

API v4 Documentation

Public Beta 1

Table of Contents

- 1. Introduction
- 2. API Description
 - 1. Authentication
 - 2. Functions
 - 1. Motifs
 - 2. Image
 - 3. Job
 - 4. JobInfo
 - 5. Result
 - 3. Data testing
 - 4. When to use which command?
 - 5. Code samples
 - 6. Various mentions/notes

1 Introduction

AlphaPicture offers an API to retrieve personalized images. Such images require preparation in order to be personalized and once this preparation has been performed by us we call those images "Motifs". Note that these motifs can only be created by our team of experts. There is now way to create them on your side.

Those motifs are then loaded into our personalization software which will, after receiving text input, render personalized images.

This document describes the fourth version of the API which is designed to scale virtually indefinitely through the use of cloud computing. Small jobs are still run on continuously available servers in order to achieve quick processing time.

This new version strongly leans on the usage of Docker and Node.JS.

1.1 Terminology

1.1.1 *motif- alternative*

In this document you will find the term "motif" as well as the term "alternative". Think of the "motif" as the physical file on our servers that holds all code and image elements that make up a motif.

That's not completely true though. In some cases the physical file holds the code and image elements for *several* alternative representations of one motif. These alternative representations are referred to as "alternatives".

Example: Motif 163 is about fireworks. But you have 3 alternatives. 163-1 is fireworks over Berlin, 163-2 is fireworks over Stuttgart, 163-3 is fireworks over Rome. The base images for those alternatives are all in the same physical file. If they were in different files, they would have a motif number of their own. Sometimes a motif has only one base image but different alternatives because of different fonts used.

Think of the combination of the motif number and the alternative number as the only unique way of describing a motif.

1.1.2 *srcRect*

srcRect is short for source rectangle. It is used whenever you want to specify which part of the base image you want to get. Or in other words: You use srcRect if you want to define a crop.

Four values are needed to define the crop: left, top, right, bottom. These are the pixel values counted from the upper left corner of the base image. The dimensions of the base image can be looked up in the motif data that is retrieved via the **/Motifs** command (parameter: "original_rect")

1.1.3 *boundingRect*

boundingRect is a special crop (srcRect) and specific to each motif-alternative. It defines the area of an image where the personalization takes places. It is specified in the same way as a srcRect, but it is "read-only". What's it for? Well, if you want to minimize data transfer, you could decide to place the full-size image in your document and retrieve only the personalized areas from the API.

It is also useful to check whether a given srcRect cuts in the personalization area.

1.1.4 *PiP – Picture in Picture*

Some motifs support the picture in picture functionality. This means that personalization can not only be achieved by changing texts in a motif, but also by uploading an own image that will then appear in the motif as well, for instance on a billboard or a TV screen. Here's an overview of all motifs that support PiP:

<https://motifs.alphapicture.com/402-sheet-view.php?sel=pip>

When choosing an image for PiP, best practice is to prepare it in the preferred size and color depth as listed on that overview page.

For more information about how to use this feature, refer to sample code 7 (see 5.7).

2 API Description

The API is REST-based and is setup to use JSON for input and output.

The next sub-sections will explain functionalities which are relevant for implementation. More features will be exposed in the near future.

2.1 Authentication

The API requires authentication in order to prevent malicious requests. Never, ever, should you expose any AlphaPicture API directly to your customers! In all cases the only program making requests to the API should be fully in your control and not exposing any functions to the outside world.

Authentication happens through two custom HTTP-headers:

Header	Contents	Used in
APIV4-Account	Your API username as provided by AlphaPicture	Motifs/Job/JobInfo/Image
APIV4-Password	Your API password as provided by AlphaPicture	Motifs/Job/JobInfo/Image

While this may seem cumbersome, this setup prevents third parties making unauthorized requests which may directly incur costs. Alternatively these parameters can be also be passed by using basic authentication if you add the query parameter basic=1 (i. e.: <https://username:password@v4.alphapicture.com/Motifs?basic=1>)

2.2 Functions

Overview: There are basically only 5 commands:

/Motifs	returns relevant information about all available motifs
/Image	instantly returns a single personalized image
/Job	requests many personalized images from many motifs and returns a job id
/JobInfo	returns the status of a job
/Result	returns the result of a job (zip file or link)

Note that the command names are case sensitive, so /JobInfo is correct while /jobinfo is not!

Below you will find a detailed explanation for each command.

2.2.1 Motifs

The **/Motifs** function exposes information about all motifs that are available to your API account. These can be public motifs or custom motifs (made especially for you).

The result consists of some basic data such as motif id, motif name or original resolution but also contains information about the typefaces used. The latter is important for performing data testing. We strongly suggest you save this motif information locally and periodically update it using the "lastmodified" key.

2.2.1.1 Motifs - Request

Type	GET
URL	http://v4.alphapicture.com/Motifs
Authentication	APIV4-Account and APIV4-Password
Body	None

2.2.1.2 Motifs - Response

The response will be a JSON array containing information about all accessible motifs.
A single motif will be used for illustration below

```
{
  "id" : 18, // This is the motif number
  "version" : 14, // Motif version
  "name" : "Gingerbread Heart", // Motif name
  "creationdate" : "2002-10-22T00:00:00.000Z", // Creation date
  "lastmodified" : "2011-11-21T00:00:00.000Z", // Last modification date
  "alternatives" : [{ // This is a list of all alternatives existing for this
    // motifs have multiple alternatives with
    // alternative number
    "alternative_id" : 1,
    "motif_id" : 18, // motif id
    "name" : null, // alternative description, null if only
    // alternative is
    "photographer" : "Michael von Aichberger",
    "programmer" : "Michael von Aichberger",
    "client" : "AlphaPicture", // This will be AlphaPicture for
    // 'public' motifs
    "programmers_note" : null, // Some motifs have text here which
    // guidelines how
    "sample_text" : { // This can be used to create sample images
      "1" : ["Merry Christmas"],
      "2" : ["with"],
      "3" : ["AlphaPicture"],
      "4" : ["xxx"]
    },
    "lines" : { // This is an array with all lines available in
      "1" : {
        "typeface" : "1", // Each line has a typeface
        "length" : "3650" // Each line has a certain
      },
      "2" : {
        "typeface" : "1",
        "length" : "3810"
      },
      "3" : {
        "typeface" : "1",
        "length" : "2800"
      },
    },
  }
}
```

```

        "4" : {
            "typeface" : "1",
            "length" : "1850"
        },
        "original_rect" : "0, 0, 3068, 4095", // Original size of the
alternative
        "bounding_rect" : "958, 1953, 2568, 3810" //
Personalization area
    },
    "typefaces" : [{ // Each motif has a set of typefaces associated with it
        "typeface_id" : 1, // Typeface have an id, this is referenced
by lines in //alternatives
        "motif_id" : 18,
        "typewidthL" : // This is a list of how much space every
character takes, // refer to data
testing section for how-to-use
"[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,129,99,0,0,0,0,231,99,0,0,
,0,247,112,175,99,0,260,223,249,211,226,240,235,255,240,220,104,0,0,0,0,191,233,340,227,2
34,320,237,292,292,217,246,298,231,266,375,311,238,237,255,244,240,255,254,238,287,234,23
8,240,0,0,0,0,0,0,212,202,194,249,213,232,241,247,178,220,220,216,252,216,188,249,244,237
,231,228,223,196,291,181,275,275,.....]",
        "symbolL" : null, // Refer to data testing section
        "charspacing" : "0.0000" // Refer to data testing section
    }
}

```

2.2.2 Image

The **/Image** function allows you to generate single images rapidly and retrieve them instantly. So one request results in exactly one rendered image.

Note that if you pass dimensions with an aspect ratio different from that of the base image, the retrieved image will be cropped out of the center of the base image with no control of how well that crops looks or even whether the personalization area is truncated.

In most cases a centered crop with not too extreme aspect ratios will look just fine, but if you want to control it yourself, you must set the `srcRect` parameter accordingly.

2.2.2.1 Image Request

Type	POST
URL	https://v4.alphapicture.com/Image
Authentication	APIV4-Account and APIV4-Password
Body	<pre>{ "MotifId": 614, // Id of the motif "AlternativeId": 2, // Alternative Id "Lines": { // Array of lines "1": "JOHN DOE" }, </pre>

```
        "Dimensions": { // Dimensions in pixels
            "w": 881,
            "h": 327
        },
        "SourceRect": { // Optional: SourceRect is for retrieving a
            // the original image
            "x1" : 1532,
            "y1" : 1353,
            "x2" : 3801,
            "y2" : 2195
        },
        "Watermark": false // Optional: set to true if image should be
        watermarked
    }
```

2.2.2.2 Image Response

Response will be a single JPG image

2.2.3 Job

The **/Job** request allows you to bundle requests for several (i. e. many) images (same or different motifs, same or different text) in one call. The response to this call is not an image, but an ID (in UUID format). Use this ID to check the progress of the job with the **/JobInfo** command and to retrieve the results (the images / link to the images).

2.2.3.1 Job Request

Type	POST
URL	https://v4.alphapicture.com/Job
Authentication	APIV4-Account and APIV4-Password
Body	<pre>{ "Images" : [{ // Array of motifs that need to be rendered "MotifId" : 4, "AlternativeId" : 1, "Template" : { // See below for explanation on template '1+2' : "Hello %firstname% %lastname%" }, "Filename" : "One_%number%", // Filename, without JPG extension "Dimensions" : { // Dimensions in pixels "w" : 1000, "h" : 750 }, "Watermark" : false } }</pre>

```

    ],
    "Callback" : "http://some.callback.com/", // HTTP callback
    called when job done
    "OutputOptions" : { // This are options related to the output
    of this job
        "OutputMethod" : "HOTLINK" // See below for options
    },
    "Data" : [{ // This is the data object which used in
    substitution, see below
        "firstname" : "John",
        "lastname": "Doe",
        "number" : 1
    }, {
        "firstname" : "John",
        "lastname": "with a long name",
        "number" : 2
    }
    ]
}

```

The body outlined above is a simplified example of what is possible in a job request. Some important parameters are explained in more detail below:

Parameters explanation

Template	<p>Images in a job are rendered based on a template. A template is an object which defines the personalized text that is rendered in an image. A motif can feature one or more personalizable lines of text. A text can spread over a single line or over multiple lines. If multiple line numbers are passed (in a '1+2+3' fashion), the text will be split on spaces and/or hyphens while trying to fill the first lines as much as possible.</p> <p>The text in an image can be static or variable (personalized, derived from a database) or a mix of both. For the personalized part, variables are used. Variables are noted with percent signs and refer to data objects which are passed in the data array. Naming of these variables is completely arbitrary – except that variable names themselves cannot contain percent signs.</p>
Filename	Filenames use the same variables as templates
Callback	When a job is done and if a callback URL has been passed this callback will be called with a JSON body. The fields JobId, Status, Key and DownloadURL will be passed back to this callback
OutputMethod	<p>Currently we have two output options:</p> <ul style="list-style-type: none"> - HOTLINK → this will output all images on a CDN (Content Distribution Network) for direct hotlinking usage. The hotlink URL will be available in JobInfo. - ZIP → the files will be zipped up on our servers and made available for downloading
Data	<p>Jobs work based on data objects. An array of data objects should be passed with the request. For every data object one image is rendered for every object in the "Images" array. Thus passing 5 different motifs with different templates and sizes and 1 data object will render these 5 motifs for this single data object.</p> <p>Data objects can contain any key/value</p>

2.2.3.2 Job Response

A JSON object in the following format will be returned:

```
{
  "JobId" : "f5601144-a6d5-4008-b4be-1c3b3437f9e9",
  "Error" : false,
  "Status" : "IN_PROGRESS",
  "Key" : "f81d4fae-7dec-11d0-a765-00a0c91e6bf6",
  "CDN" : "http://cdn.alphapicture.com/f81d4fae-7dec-11d0-a765-00a0c91e6bf6/"
}
```

Good practice is to save JobId along with the request that you have stored somewhere, this id can later be used to fetch the status of a job and retrieve other information about it. CDN links are not available for jobs which do not have the "HOTLINK" OutputType.

2.2.4 JobInfo

The **/JobInfo** function allows to retrieve information such as status about a specific job.

2.2.4.1 JobInfo Request

Type	POST
URL	https://v4.alphapicture.com/JobInfo
Authentication	APIV4-Account and APIV4-Password
Body	<pre>{ "JobId": "f5601144-a6d5-4008-b4be-1c3b3437f9e9" }</pre>

2.2.4.2 JobInfo Response

A JSON object in the following format will be returned:

```
{
  "JobId" : "f5601144-a6d5-4008-b4be-1c3b3437f9e9",
  "Error" : false,
  "Status" : "IN_PROGRESS",
  "Key" : "f81d4fae-7dec-11d0-a765-00a0c91e6bf6",
  "CDN" : "http://cdn.alphapicture.com/f81d4fae-7dec-11d0-a765-00a0c91e6bf6/"
}
```

Good practice is to save the JobId along with the request that you have stored somewhere, this id can later be used to fetch the status of a job and retrieve other information about it. CDN links are not available for jobs which do not have the 'HOTLINK' OutputType

2.2.5 Result

The **/Result** function allows you to retrieve resulting images or a zip file from a specific job.

2.2.5.1 Request

Type	GET
URL	https://v4.alphapicture.com/Result/{key}/{filename}
Authentication	None
Body	None

2.2.5.2 Response

Requesting with just a key (see **/JobInfo** or response from Job for retrieving a key) will result in downloading a zip file. If a file name is specified then this specific file is returned as JPEG.

3 Data testing

One important part of integrating the AlphaPicture API is understanding that not all text will work in any motif. There are constraints on a number of things:

- Characters available in a motif
Imagine fonts that were hand-made which might not contain all eastern European characters
- Length of text in a line
In order to prevent text becoming unreadable or looking unrealistic we enforce constraints on the maximum length a line can be

We always do a server-side check of all these constraints and will notify your system when an impossible request is encountered, but in order to deliver end users a smooth experience we strongly advise you to implement our data testing algorithm in any web frontend for end users.

3.1 Step 1 – Checking for character availability and possible substitution

In order to check whether a character is available within a certain motif you should follow the following procedure:

- Identify the unicode position of the character you want to verify (i. e. "€" = unicode 128)

- Identify the line in which the text is entered. As any line in a motif can use a different typeface, check which typeface is used in that particular line. This is done by looking at the "lines" definition in the motif information retrieved from the API. There you will have the typeface-to-line allocation.
- Within any typeface definition there is a field called "typewidthL" which contains a rather large array. This array holds the virtual character widths of all characters supported by this typeface. We will need those values for the calculation of the total text width. Note that in this document, "width" and "length" of a text are used interchangeably, always referring to the width of a text, because we do not just count the number of letters in a line. Instead, we add the individual widths of the characters plus a value for the space between characters plus in some cases additional calculations for kerning. For example: If you want to look up character 128 then check the 128th position in this list. If at this position there is a number greater than zero this means the character is supported, if there is a width of 0 then this character is not supported. **Note that the indexing of that array starts with the value of 1, not with 0 as in many programming languages.** Thus, the index is identical to the unicode of a character.
- If a character is not supported then you should check if it has viable alternatives available by checking the "substitution_list.json" file. This contains a field for every character that can be substituted which holds an array of possible substitutions.

For example, if a typeface supports only uppercase letters and you pass lowercase text, you don't want the motif to show now text at all. Instead you expect your lowercase text to be converted into uppercase text. Or, if a motif doesn't support characters with accents, like "é" you might want those to be converted to the base characters without accent. In substitution list there are even more complicated suggestions for substitutions. For example: If your text contains a lowercase o umlaut "ö", it might be converted to "oe". If "oe" is not possible (in the case of the typeface being uppercase only), "Ö" is tried. And if that's not possible either, "OE" is tried. If there actually is a substitute in the list, you should verify that the suggested substitue character is available in typewidthL as above. If not, try the next substitue. You might do this as long as you have either found a viable substitute or until there is no suggestion for substitution left. If you found a substitution, everything will be ok, otherwise it is not possible to render this motif with that specific text.

- Please note the use of a special character (unicode 164): In some motifs we allow the users to enter a heart symbol – in very rare cases even more symbols. You will find the information whether a typeface supports the heart symbol in the field "symbolL" of the motif information. Normally, unicode 164 represents a "currency symbol" – hardly used by normal people. So we decided to hijack this unicode number and reuse it for the heart symbol.

3.2 Step 2 – Checking if width has been exceeded

Once you have identified the width of all characters on a line (by looking them up in the `typewidthL` list) and performed possible substitutions, it is time to add up all those values and check the sum against the "length" property of the specific line. If charspacing (the space between characters) is not 0 (it can have positive or negative values), you have to take this value into account as well. Remember that if n is the number of characters, $(n - 1) * \text{charspacing}$ is to be added. If the total length of your line is less than the allowed maximum you are ready to render images.

4 When to use which command?

As already stated, you have 2 methods of retrieving images:

With the **`/Image`** command and with the **`/Job`** command.

The **`/Image`** command requests and retrieves 1 image only. The image is immediately returned after it has been rendered, which normally should be within 1 to 3 seconds (different motifs/output sizes/text lengths etc. take longer or less to render).

The **`/Job`** command on the other hand allows you to request multiple images from multiple motifs while passing multiple data objects (i. e. text definitions for the personalization).

So, when would you choose one command over the other?

If you only want to render 1 image, use **`/Image`** by all means. This is the quickest way, as you don't have to check the status of the request. If you have some more images to render (more than 1, but not "many"), you can still use **`/Image`** commands. You don't have to pass them one by one, you can send a few (around 5) of them at the same time.

If you want to generate more than 5 images at the same time, use the **`/Job`** command. For instance for a calendar, you typically want to retrieve 13 images. Request them in 1 **`/Job`** command! Of course, if you have a list of names and want to create 1 calendar for each name, you still need only one request. For a calendar with 100 names you will issue 1 **`/Job`** command requesting $100 * 13 = 1,300$ images.

When using the **/Job** command, the quickest way to get the results is to pass a callback URL. Thus you don't have to issue many **/JobInfo** requests to know when the job is done. Instead, your code just waits for the callback. You won't lose any time.

If you use the **/JobInfo** command, don't pass it too frequently. 2 seconds intervals should do.

5 Code samples

In order to use the included code samples you will have to install Node.js - if it is not already existing on your system.

See <http://nodejs.org/> for further reference.

You should unzip the samples to a folder and then from within this folder run "npm install request" in order to fetch a dependency. Then you can run the samples by running "node 1_motifs.js" from the command line. You will however need to enter your username/password in each sample file for them to run correctly.

5.1 Retrieve the motif list and save it locally

File: 1_motifs.js

This sample shows how to query the AlphaPicture API for all motifs available to you. The response is saved in the file motifs.json.

5.2 Retrieve a single image

File: 2_Image.js

This sample retrieves a single image and stores it in the output folder with a prefix of 2_. Make sure the folder exists. Feel free to adjust parameters. This functionality should be used if you want to quickly show a preview of a single image.

5.3 Retrieve a single image for each available motif to build a gallery

File: 3_gallery.js

This sample uses the database saved by sample 1 in order to fetch all files with their stored sample text.

5.4 Retrieve a job with 5 names for 1 motif

File: 4_Job.js

This sample retrieves 5 images (packed in a zip file) from a single motif with a dataset of 5 names and shows how to request the current status of a job.

5.5 Retrieve a job with 1 name for 3 motifs

File: 5_job.js

This sample creates 3 images from different motifs with the same text and shows the CDN link to download them.

5.6 Job with callback

File: 6_job.js

This sample runs a dummy job with a callback attached to <https://webhook.site>, you should set up your own webhook.site unique URL callback in order to see how this works.

5.7 Retrieve an image with PiP (Picture in Picture)

File: 7_pip.js

This sample shows how to upload your own image and use it for the personalization a motif. Please note that most motif do not support this feature. Refer to „pip_count“ in the motif parameters to check whether a motif supports PiP.

6 *Various mentions/notes*

- When making requests towards the API always be sure to include a Content-Type: application/json header
- This is a first public release of the APIv4, if you encounter any problems, please do let us know!
- There is a legacy syntax for API requests. It is basically a http:// request with cgi parameters. It was used in v1 and v2 of our API and is still supported in v4 for backward compatibility. As it is easier to set up, there might still be use cases where you would prefer the old over the newer and more powerful json syntax. If you feel you have such a use case, please contact us for details.
- If you have any feature requests and/or improvement suggestions then please report them to m.v.aichberger@alphapicture.com
- If you require any support please contact support@alphapicture.com